ISI Reprint Series
ISI/RS-93-416
August 1993

Using Prospero to Support Integrated Location-Independent Computing

B. Clifford Neuman, Steven Seger Augart and Shantaprasad Upasani

ISI/RS-93-416
August 1993

University of Southern California
Information Science Institute
4676 Admiralty Way, Marina del Rey, CA 90292-6695
310-822-1511

This Research was supported in part by the National Science Foundation (Grant No. CCR-8619663), the Washington Technology Centers, Digital Equipment Corporation, and the Advanced Research Projects Agency under NASA Cooperative Agreement NCC-2-539. The views and conclusions contained in this paper are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of any of the funding agencies.

Using Prospero to Support Integrated Location-Independent Computing

B. Clifford Neuman Steven Seger Augart Shantaprasad Upasani

Information Sciences Institute
University of Southern California

Abstract

As computers become pervasive, users will access processing, storage, and communication resources from locations that have not been practical in the past. Such users will demand support for location-independent computing. While the basic system components used might change as the user moves from place to place, the appearance of the system should remain constant.

In this paper we discuss the role of, and requirements for, directory services in support of integrated, location-independent computing. We focus on two specific problems: the server selection problem and the user location problem. We present solutions to these problems based on the Prospero Directory Service. The solutions demonstrate several unique features of Prospero that make it particularly suited for support of location-independent computing.

1 Introduction

As the use of computers becomes pervasive the distinction between computer networks and computer systems will blur. In the ideal world, users will think of a computer network and the systems connected to it as a single system, rather than as a collection of systems connected by networks. Users will not want to use a different system each time they change their location. Although users will want to see a single system, they won't want to see the same system as every other user. Each user will want a system that is tailored to his or her particular needs.

This paper begins with a discussion of the characteristics of and requirements for what we call pervasive computing. We examine two problems that arise in such systems, the server selection problem and the user location problem, discussing the role played by a distributed directory service in their solution. In so doing, we describe the Prospero Directory Service, highlighting important features, and describing how it can be used to solve these problems.

2 Pervasive Computing

Pervasive computing combines aspects of ubiquitous computing with the integration of information and resources from many sources, within a single system tailored to the needs of a particular user. Whereas the focus of ubiquitous computing has been on the devices and the communication infrastructure, allowing the use of large and small computing devices from many locations, the focus of pervasive computing is on mechanisms that allow the pieces to be tied together to form a coherent whole. The two areas are not disjoint; each includes the other, only the perspective is different.

There are several characteristics to pervasive computing that place new demands on system organization and structure. One of the primary characteristics is mobility. The term mobility applies even to systems that don't support wireless communication; it is the mobility of users that is critical to pervasive computing. Users interact with the system from more than one location. We already see this on large university campuses where students log in from public terminal clusters.

Proceedings of the Symposium on Mobile and Location-Independent Computing, Cambridge, MA, August 1993.

The use of portable computers while traveling provides another example. In the future, users will be able to interact with the system through whatever I/O device is within reach as they travel from location to location.

A second characteristic of pervasive computing is scale. The number of objects and services to be managed can easily overwhelm the user, the geographic expanse of the system adds constraints to be considered when selecting servers, and the lack of a single organization that controls the system makes organization of these resources difficult. These characteristics can be addressed in part through support for customization. Users should be able to choose the resources and objects of interest, and treat the selected resources as a single system [5].

The directory service will play a critical role tying together the components of future systems. The requirements for such a directory service are greatly affected by the scale of the system, and the mobility of its users. One of the biggest problems to be addressed is support for transient information. The transient information in such a system comes in two forms: first, the choice of servers for certain operations may change as the user moves from location to location; and second, information about users and other mobile objects needs to be maintained.

3 The Server Selection Problem

We begin our discussion by considering the server selection problem. Selecting resources for use in a centralized system is straightforward: users choose from among the resources available and select defaults which rarely change. In traditional distributed systems, the selected resources are then located through remote name maps or directory services such as Sun's Network Information Services (formerly known as Yellow Pages) [6] and Hesiod [3].

The mobility of users complicates server selection; the user's choice of resources will often vary according to location. For example, while at home a user might want to use a printer at home and while at work, one down the hall. While traveling the user might want output faxed to the hotel's front desk, but only if there is no per-page charge for incoming faxes. Alternatively, a user might want output sent to the printer at home so that it is waiting upon return.

It should be possible for users to specify defaults in such a way that the binding is determined dynamically, when the service is needed, and based on a combination of user specified factors (e.g., cost and reliability), application requirements (e.g., support for PostScript), and transient factors (e.g., load and proximity).

3.1 Using Prospero to Solve the Server Selection Problem

With Prospero, users define a virtual system which, among other things, specifies the mapping of names to servers. This mapping is used to select the servers used by applications. Figure 1 shows how a virtual system is represented using Prospero. In the figure, the link labeled ROOT is a reference to the root of the user's file system through which the user sees the same files regardless of the location from which logged in. The Prospero File System is described elsewhere [4]. The directory referenced by the SESSIONS link identifies the locations from which the user is logged in and is described in Section 4.

In this virtual system, the server selection criteria are encoded in the CONFIG/SERVERS directory. In a traditional directory service, such a mapping would not provide the flexibility that is needed for pervasive computing. However, several features of the Prospero Directory Service allow the mapping to be determined dynamically. These features include virtual system aliases, union links, and filters.

3.1.1 Virtual system aliases

Prospero maintains several virtual system aliases that provide well defined starting points from which names may be resolved. These aliases end with the string #: and identify the virtual systems associated with the user, the home processor for the current login session[1] (the session), the processor on which an application is running (the platform), and open file descriptors.

[1] The virtual system for the session is associated with the workstation or I/O device through which the user is interacting with the system.

Proceedings of the Symposium on Mobile and Location-Independent Computing, Cambridge, MA, August 1993.

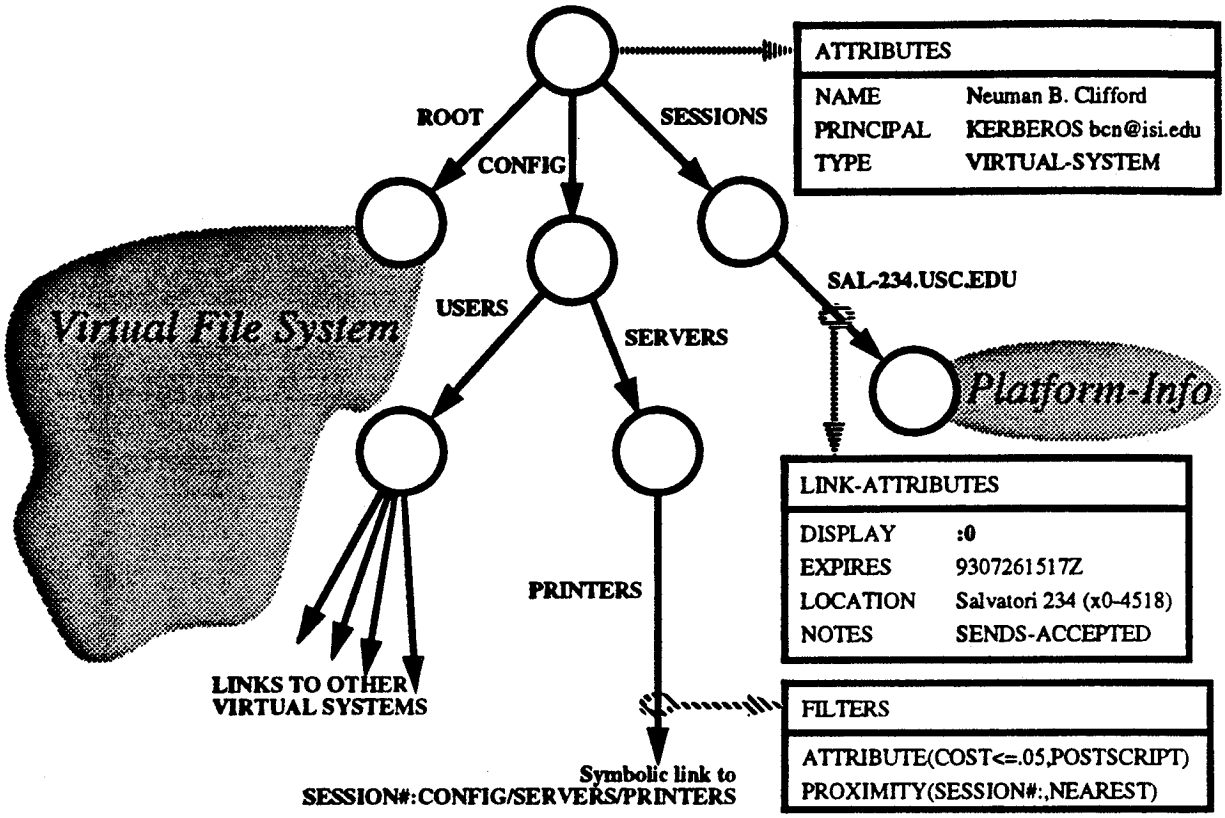

Figure 1: Structure of a Virtual System

In the figure, the directory CONFIG/SERVERS/PRINTERS defines the printers available to the user. If the user wanted to use the same printers always, the link would refer to a directory that explicitly named the printers to be used. Here, the user chose to define the available printers as a symbolic link to the directory of printers for the session. Because the target of the symbolic link begins with the string SESSION#: the rest of the name is resolved in the virtual system for the session, which is reachable through the node for the login platform and is part of the cloud labeled *Platform-Info* in the figure.

3.1.2 Union links

A user who wanted to choose from among the printer at home, the one at work, and those nearby, could use a union link to create a dynamic view of available printers. The CONFIG/SERVERS/PRINTERS link would refer to a directory with references to the printers at the user's home and office. That directory would include a symbolic union link to the directory of printers for the session. The resulting directory would appear to contain the union of the two directories.

3.1.3 Filters

Users should be cautious when allowing the system to select servers for use in unfamiliar locations. Business travelers have been taught this lesson by unscrupulous alternative telephone operator services that charge exorbitant fees for long distance phone calls. Using Prospero, users are able to constrain the selection of servers by applying a filter to a directory. A filter is a function attached to a link that modifies the result of a directory query. In figure 1 the user applied the **attribute()** filter to restrict the selection of printers to those charging no more than 5 cents per page, and supporting

PostScript. Of the printers selected by the **attribute()** filter, the **proximity()** filter selects the closest, provided that the implementer of the **proximity()** filter devised an appropriate heuristic.

A filter can also change the order in which servers are presented to the user setting the COLLATION-ORDER attribute of the links it returns. This allows filters to sort servers according to user specified criteria.

Prospero supports two kinds of filters: loadable and predefined. Loadable filters are dynamically loaded and executed during name resolution (their present implementation is not portable and presents security problems). Predefined filters are compiled into the name resolution library. Predefined filters have registered names and must be present in the name resolver (or on the server for some filters). Predefined filters must be widely supported to be useful; therefore they usually provide general operations such as selection based on the values of attributes.

3.1.4 Supporting mobile platforms

Our discussion so far has assumed that a stable platform exists for each session, and that a server directory has been defined for each platform. The discussion has ignored problems related to mobile platforms. The mobile platform problem can be addressed by another layer of indirection: the platform might itself use a filter in the definition of its server directory. Such a filter might locally broadcast a query in search of nearby platforms willing to provide such a directory. The resulting directory would include links for each server that responds.

3.1.5 Presenting selections to the user

When selecting a server, an application can indicate additional constraints to be applied by specifying additional filters. Once all filters have been applied, if the result is a single link, the referenced server can be used. For example, the NEAREST argument to the **proximity()** filter in the figure might result in the automatic selection of the nearest printer.

If the directory contains multiple links after the application of all filters, then the user could be prompted through a dialog box to select one. This might occur if the argument to the **proximity()** filter were specified as NEAREST 5, resulting in the return of the 5 nearest printers. Once selected by the user, the choice should remain in effect until some event specified by the user, such as a change in location, or a change in the list of available servers.

4 The User Location Problem

The user location problem is a second problem that illustrates the requirements imposed on a directory service by user mobility. In centralized systems, locating an active user is easy; users are either logged in, or they aren't. If logged in, the system records the the terminal in use and makes this information available to applications such as `finger`, `write`, and `send`. In a distributed system, the problem is considerably more complex.

A common approach to the user location problem is to replicate the data on all hosts. This is the approach taken by `rwho`. Systems broadcast the names of the users that are logged in, and others store this data locally where it can be searched by the user or application. The primary drawback of this approach is that it doesn't scale very well. A second concern is privacy; users might not want others to know where they are logged in, or that they are logged in at all.

A different approach is taken by the Zephyr [2] system at MIT's Project Athena [1]. Zephyr provides a single database that may be consulted when a user's location is needed. This database is replicated for reliability (technically, Zephyr provides a notification service that relies on this database, but it is the database that is of interest here). Zephyr addresses privacy because each user decides whether to register a session with Zephyr, and to what classes of other users the login location is to be visible. Zephyr does not provide fine-grained control over access to user location data. Though suitable for a large campus, the use of Zephyr as a user location database does not scale across administrative domains.

4.1 Using Prospero to Solve the User Location Problem

A third approach is to use a directory server to store user location information. Such a directory server would have to tolerate frequent updates. If a user is to be able to specify the principals who can obtain his or her location, then support for fine-grained access control is also necessary. Finally, it must be possible to authenticate both updates and queries.

The Prospero Directory Service already maintains information about a user's virtual system. This information has been extended to include information about login sessions. At login, an entry is added to a list of sessions, and at logout the entry is removed. By associating an expiration time with the entry, it is possible to detect sessions that are not properly terminated. By placing an access control list on the list of sessions, a user can specify on a per-principal basis the individuals to which the session is visible. The SESSIONS link in figure 1 points to the directory that maintains a list of sessions.

Prospero is a distributed directory service, and it is likely that user information will be distributed across a large number of systems, maintained by multiple organizations. Each directory server enforces its own access control. As such, if a user's directory information is stored on a trusted server for the user's organization, the user's privacy depends only upon the security of that server.

Through its support for customization, Prospero allows a user to define a set of colleagues, and the name used to refer to each. This set initially contains the names of other users in the local organization, with users beyond the organization named hierarchically based on the name of the organization to which they belong. Users can define their own short names for remote colleagues, after which they are referred to no differently than if they were local. This is represented in figure 1 by the CONFIG/USERS directory. In fact, this customization mechanism allows a user to define the set of colleagues considered local for use by `finger`; when run with no arguments it displays the locations of users currently active and identified as colleagues by the user.

When using modified versions of commands such as `send`, `talk`, or `finger`, the name of the target (another user) is specified relative to the CONFIG/USERS directory. The command consults the directory server to determine the location of the target user, and if found, performs the requested operation.

5 Establishing a Session

The solutions to both the server selection and the user location problems depend on several operations being performed when a new login session is established. This section describes what happens when a user logs into a system supporting the Prospero login program. In this discussion, the platform is the processor on which the application (in this case the login program) runs.

When a user attempts to log in to a system running the Prospero login, the system uses the name of the user to find the user's virtual system. For local users the virtual system is found in the directory of virtual systems for the local site. For remote users the virtual system is found starting from a directory that identifies other sites. The PRINCIPAL attribute associated with the user's virtual system is examined and used to select an appropriate authentication method. The user is authenticated and the login program determines whether the user is authorized to use the resources of the platform.

Once logged in, the namespace is defined by the user's virtual system and the virtual system alias SESSION#: is defined as the virtual system of the platform on which the user logged in. Next, if the user was suitably authenticated, an entry is made in the SESSIONS directory of the user's virtual system recording the platform, the login time, an expiration time, and other information associated with the session. During a session, name resolution occurs in the name space defined by the user's virtual system. When the session is terminated, the entry in the SESSIONS directory is removed.

6 Status

The Prospero Directory Service has been available since December 1990 and has been used to support the integration of information services on the Internet. The recent release of Version 5 of Prospero allows it to be more easily integrated with other applications. This paper described some of the ways that Prospero can be used to support integrated location-independent computing. Prospero presently provides the basic mechanism needed to address the problems discussed in this paper. We have started work on the user location problem as part of the implementation of a Prospero-based login program. We have not yet started work on the server selection problem, but plan to do so in the near future. To find out more about Prospero, or for directions on retrieving the latest distribution, send a message to info-prospero@isi.edu.

7 Summary

Pervasive computing places new demands on directory services. Among the demands is a need to support transient data, fine-grained authorization for queries and updates, and the ability to support dynamic (functional) bindings from names to servers and objects. These requirements are met by the Prospero Directory Service, which can play a role in support for truly integrated, location-independent computing.

Acknowledgments

Many individuals contributed to the design and implementation of Prospero. Ed Lazowska, John Zahorjan, David Notkin, Hank Levy, and Alfred Spector helped refine the ideas that ultimately led to the development of Prospero. Kwynn Buess, Steve Cliffe, Alan Emtage, George Ferguson, Bill Griswold, Sanjay Joshi, Brendan Kehoe, and Dan King helped with the implementation of Prospero and Prospero-based applications. Celeste Anderson, Sio-Man Cheang, Gennady Medvinsky, Santosh Rao, Eve Schooler, and Stuart Stubblebine commented on drafts of this paper.

References

[1] George A. Champine, Daniel E. Geer Jr., and William N. Ruh. Project Athena as a distributed computer system. *IEEE Computer*, 23(9):40–51, September 1990.

[2] C. Anthony DellaFera, Mark W. Eichin, Robert S. French, David C Jedlinsky, John T. Kohl, and William E. Sommerfeld. The Zephyr notification service. In *Proceedings of the Winter 1988 Usenix Conference*, pages 213–219, February 1988.

[3] Stephen P. Dyer. The Hesiod name server. In *Proceedings of the Winter 1988 Usenix Conference*, pages 183–189, February 1988.

[4] B. Clifford Neuman. The Prospero File System: A global file system based on the Virtual System Model. *Computing Systems*, 5(4):407–432, Fall 1992.

[5] B. Clifford Neuman. *The Virtual System Model: A Scalable Approach to Organizing Large Systems.* PhD thesis, University of Washington, June 1992. Department of Computer Science and Engineering Technical Report 92-06-04.

[6] Sun Microsystems. *Yellow Pages Protocol Specification*, February 1986. In Networking on the Sun Workstation.

This research was supported in part by the National Science Foundation (Grant No. CCR-8619663), the Washington Technology Centers, Digital Equipment Corporation, and the Advanced Research Projects Agency under NASA Cooperative Agreement NCC-2-539. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any of the funding agencies. Figures and descriptions in this paper were provided by the authors and are used with permission.

The authors may be reached at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, USA. Telephone +1 (310) 822-1511, email bcn@isi.edu, swa@isi.edu, prasad@isi.edu.